FTD-TT- 62-1496

TRANSLATION

COMPILING AND INTERPRETING SYSTEMS FOR USE OF STANDARD PROGRAMS FOR THE BESM COMPUTER OF THE ACADEMY OF SCIENCES USSR COMPUTER CENTER

By

V. M. Kurochkin

FOREIGN TECHNOLOGY DIVISION

AIR FORCE SYSTEMS COMMAND

WRIGHT-PATTERSON AIR FORCE BASE

OHIO

FTD-TT- 62-1496/1+

UNEDITED ROUGH DRAFT TRANSLATION

COMPILING AND INTERPRETING SYSTEMS FOR USE OF STANDARD PROGRAMS FOR THE BESM COMPUTER OF THE ACADEMY OF SCIENCES USSR COMPUTER CENTER

BY: V. M. Kurochkin

English Pages: 52

S/5882

FTD-TT- 62-1496/1+2 Date 9 May 19 63

S

Akademiya Nauk SSSR
Vyshislitel'nyy Tsentr

KOMPILIRUYUSHCHAYA I INTERPRETIRUYUSHCHAYA SISTEMY ISPOL'ZOVANIYA STANDARTNYKH PROGRAMM DLYA MASHINY BESM-2 VTs AN SSSR

Vychislitel'nyy Tsentr AN SSSR

Moskva - 1961

Pages: 1-57

FTD-TT-62-1496/1+2

TABLE OF CONTENTS

INTRODUCTION

In solving problems on automatic computers, it is frequently necessary to make use of standard routines. The existence of a library with a large set of standard routines and a convenient way of using them can substantially simplify and ease the process of preparing problems for solution. Sometimes convenience in making use of standard routines is achieved owing to the fact that the computer includes a special memory device into which the various standard routines may be loaded. For an entire series of reasons, such a solution to the problem may not be considered convenient. One of these reasons is that many computers, including the BESM-2, have no such memory device.

Below we shall describe a method for making use of standard routines with the BESM-2 automatic digital computer.

The basic information that the programmer must know is presented in §2-5. In working with the computer, it is useful to be acquainted with §6. In writing the standard routines that should be included in the library, it is necessary to allow for the requirements discussed in §7. Sections 9 and 10 give the compiling and interpreting programs, their flow charts, and descriptions.

The basic principles of the system described here for using standard routines were developed at the end of 1958. While writing the programs, we became acquainted with the interpreting system developed by Professor M.R. Shura-Bura at the OPM of the Institute of Mathematics of the Academy of Sciences USSR, and this is reflected to some degree in the structure of the interpreting program described in

§10. The first versions of the interpreting and compiling systems were written and developed for the BESM-2 computer at the Computing Center of the Academy of Sciences USSR beginning in May of 1959. The programs were then subjected to several technical modifications due to changes introduced in the instruction system of the BESM-2 machine. The present version corresponds to the system of instructions recommended by a conference of representatives from several interested organizations for all BESM-2 computers, and introduced on the BESM-2 machine at the Computing Center of the Academy of Sciences USSR 1 February 1961.

We suggest that the reader acquaint himself with the description and instruction system of the BESM-2 computer.

§1. COMPILING AND INTERPRETING

The system described for making use of standard routines provides for carrying out the following operations automatically:

– calling for the standard routines mentioned in the main program;

– calling for those standard routines that are not used directly in the main program (and are not mentioned), but which are needed for the execution of standard routines already called;

– storage of standard routines in free locations of the operating (main) storage device;

– processing of the standard routines, carried out so that the standard routines called for can be executed at the locations in the operating storage set aside for them (this modification consists in appropriate replacement of the internal addresses of the standard routines);

– organization of the linkage between the main program and the standard routines and among the standard routines themselves, with the aid of appropriate control-transfer instructions.

In the system described, two methods may be used in working with standard routines. With the first of these, carried out by the compiler, all standard routines needed for execution of the main program are loaded into the operating storage device from the very beginning (we discuss a more flexible application of the compiler in §5). The compiler processes the routines, and as a result the final object program, ready for execution, is formed in the operating memory. During execution of the main program, there is no need to use the compiling program, as a rule (see §§ 3 and 5).

With the second method, the auxiliary (interpreting) program should at all times be located in the operating storage device while the main program is being executed. Its role amounts to the following: at the instant that there is a call for any standard routine, the interpreter prepares it for execution. The appropriate standard program is called from a magnetic drum and placed into a block set aside for this purpose in the operating storage. Several standard routines may be located simultaneously in this block – it all depends upon their length and the size of the block. If the next standard routine cannot fit into the free location in this block, it "blocks" the old (previously called) standard routines. All of this naturally delays the operation of the main program. If, however, there occurs a new call for a standard program that is already located in the block set aside, only a small portion of the interpreter is executed, and the delay is decreased. For repeated execution of the instructions calling for standard programs located in the isolated block (for example, in looping), the interpreter does not participate in the operation, in general, and it causes no delay whatsoever.

The choice of a particular method of operation also depends on the standard routines used in the problem (not all of them permit operation with the interpreting program), as well as on the wishes of the programmer. The form of the call for standard routines does not depend on which of the programs - the compiler or the interpreter – executes the operation. The precise form of the call is shown in the catalog (or in the descriptions) of the standard routines. Here we shall only give the general form of such calls. Each standard program is assigned some number N. In order to call for a standard routine with the number N, it is necessary to write the instruction

$$\chi \; \boxed{77}\boxed{abcd}\boxed{8}\boxed{3777}\,.$$

in the main program.

It is possible that it will first be necessary to place certain quantities in previously-assigned locations; it is also possible that the results will be placed in certain specific locations.

Depending upon which standard routine is used, additional information may be given in one or several succeeding lines (and in the octal digit positions cd of the instruction χ).

In §5 we discuss the role of the ab digit positions of the instruction χ.

§2. STORAGE OF STANDARD ROUTINES, COMPILING AND INTERPRETING PROGRAMS

Any external storage devices may be used to store standard routines, i.e., magnetic drum, magnetic tape, or punched cards. The programmer need not know the precise arrangement of the standard routines used by him (for example, their position on drums or the number of the zones on magnetic tapes); he should know only the form (drum, tape, or punched card) of the external memory device in which the standard routines of interest to him are stored.

Standard routines located in an external memory of any type may be used with a compiler, while only standard programs located on magnetic drum may be used with the interpreter.

For each call to the compiling program (and there may be several, as discussed in more detail in §5) no more than 22 standard routines may be placed in the operating memory (from magnetic drums, punched cards, or magnetic tape); here it is possible to use no more than one magnetic tape with standard routines (up to 63 standard routines may be written on a single tape); in this case, the magnetic tape should be rewound to the beginning.

The compiling and interpreting programs are written on a magnetic drum; both these programs and the standard routines written on drums occupy locations outside the 00000-21777 range of addresses set aside for solving problems.

§3. SIMPLEST APPLICATION OF COMPILER

In the simplest case, the compiling program executes the following operation: it examines the entire main program (or a previously specified portion of it) and places all standard routines mentioned in the main program into a block set aside for this purpose in the operating storage; it naturally performs the necessary modification of the standard routines and the main program is linked with the standard routines; in addition those standard routines not used directly in the main program but needed for the execution of other standard routines already called are automatically called for.

In the main program, each call for a standard routine should take the following form

$$\chi \; \boxed{77} \boxed{00cd} \boxed{N} \boxed{3777}\, .$$

where N is the number of the standard routine, shown in the description or in the catalog of standard routines. Prior to a call for a standard routine, the computer should operate under central control.

The preceding (..., $\chi - 1$) and succeeding ($\chi + 1$, ...) instructions, as well as the two lowest-order octal numbers c, d in the first address of the call instruction (χ) are written in accordance with the information in the description of the given standard routine or in the catalog (as a rule, c = d = 0, and the necessary changes are accomplished in the preceding and succeeding instructions, or additional information is given).

In addition, it is necessary to locate instructions calling for the compiler in the main program. These instructions should be executed just once before the first call for a standard routine is encountered; however, they should be executed after the entire main program has been loaded into the operating memory. These instructions may, for example, be located (and executed) at the very beginning of the main program.

Instructions calling for the compiler take up four lines and have the following form

$L + 1$	30	0102	3001	0016
$L + 2$	31	–	0001	0005
$L + 3$	77	α_H	α_K	0001
$L + 4$	n	β_H	β_K	A

.

Here α_N, α_K are the beginning and end (i.e., the addresses of the first and last locations) of the main program, or that section of the main program in which all calls for standard routines are contained;

β_N, β_K are the beginning and end of the block set aside by the programmer for storing the standard routines;

<u>n</u> is the number of standard routines loaded from punched cards;

A is the address of the first location in the block A-A + 0157 of the 112 (160 in the octal system) locations in which the compiler itself is located.

The following remarks must be made:

1. The blocks α_N-α_K, β_N-β_K, A-A + 0157 should not overlap.

2. Within the block α_N, α_K all lines of the form

77	~	~	3777

should indicate a call for standard routines (~ indicates that any numbers may be located in the corresponding digit positions). Generally speaking, it is impossible to use these lines as constants, since the third address (3777) in these lines will change. If it is necessary to use a constant of the form

77	~	~	3777

in the program it should be located outside the block α_N, α_K. Numbers that might take this form when there is a call for the compiler should also be placed outside this block.

3. The length of the block β_N, β_K should be at least two locations greater than the sum of the lengths λ_i of all standard routines that will be placed in the operating memory:

$$\beta_K - \beta_N \geq \Sigma \lambda_i + 1.$$

4. Standard routines punched into cards are loaded into the computer when there is a call for the compiler.

If more than one standard routine is taken in from punched cards, the selected sets of cards with the standard routines may be arranged in any order with respect to each other.

If the main program provides for information to be fed in from punched cards prior to (or following) a call for the compiler, the sequence in which the cards are arranged (in particular, punched cards with standard routines) should naturally correspond to the order of introduction.

5. The compiling program itself operates in locations A-A + 0157 only at the instant that it is called for. Following this, the block is freed, and may be used for other purposes (for example, for loading and storing new information, for storing new results, etc.).

6. After execution of the compiler, control is transferred to location L + 5.

7. When the compiler is being executed, locations in the operating memory with addresses from 0001 to 0017 are used as the operating locations.

§4. SIMPLEST APPLICATION OF THE INTERPRETING PROGRAM

Not all of the standard routines contained in the library can be used with the interpreting program. The descriptions and the catalog of standard routines show for each routine whether or not it may be used with the interpreter (one of the basic requirements is that the standard routine be stored on magnetic drum).

In working with the interpreter, it is at all times stored in the operating memory, and occupies locations from 3665 to 3777. Directly before it is located the block in which the standard routines are stored. The length of this block may be less than the sum of the lengths of all standard routines used in the main program. In this case, the standard routines being executed will replace each other in some sequence.

Naturally, the operation of the program as a whole is delayed. From this viewpoint, it is desirable to select the length of this block so that all standard routines for the innermost loop can be stored in it simultaneously. It is sufficient, for example, to satisfy the following condition

$$\gamma \leq 3663 - \Sigma \lambda_i - m,$$

where γ is the beginning of the standard-routine block; λ_i are the lengths of all standard routines for the innermost loop; m is the number of standard routines used in the main program.

It may turn out that there is a call in the basic program for some standard routine that in turn contains a call for another standard routine, which calls for a third, etc. In this case, it is necessary to set aside for the routines a block long enough to contain this entire chain of "subordinate" standard routines simultaneously. It is enough, for example, to satisfy the condition

$$\gamma \leq 3663 - \Sigma \lambda_i - m,$$

where $\Sigma\lambda_i$ is the sum of the lengths of the standard routines contained in this chain, while m is defined as before.

It is necessary to pay especial attention to this limitation since no automatic stop (halt) is provided for in the interpreting program in this case.

Finally, we should note that it is possible to use any standard routines located on magnetic drum with the interpreter provided that we set aside a block in which they can all be placed simultaneously,

$$\gamma \leq 3663 - \Sigma \lambda_i - m,$$

where $\Sigma\lambda_i$ is the sum of the lengths of all standard routines required by the main program.

In working with the interpreter, a call for standard routines takes precisely the same form as it does in working with the compiler (it is described in §3), the only difference being that the two highest-order digits a, b of the first address in the call command

77	abcd	[illegible]	3777

may take on any values.

Prior to execution of the main program, the interpreting program should be called into the operating memory with the aid of the instructions

$L+1$	30	0102	3222	0132
$L+2$	31	–	3645	3675
$L+3$	77	γ	–	3645

where γ is the beginning of the block set aside for the standard routines and the interpreting program (its end is at 3777).

Comment: 1. The value of γ should be above 2000. If it is not indicated ($\gamma = 0$) or improperly indicated ($\gamma \leq 2000$), it should be assumed to equal 3400.

2. In working with the interpreter, the third address (3777) of the instructions calling for standard routines is replaced (as in

working with the compiler).

3. After calling (instructions L + 1-L + 3) and "adjusting" the interpreter, the program transfers control to location L + 4.

§5. GENERAL COMMENTS ON THE OPERATION OF THE COMPILING AND INTERPRETING PROGRAMS

In the main program, a call for standard routines will always take the form

$$\chi\ \boxed{77}\boxed{abcd}\boxed{N}\boxed{3777}.$$

If this instruction falls within the block α_N-α_K used by the compiler (when it is called), i.e.,

$$\alpha_N \le \chi \le \alpha_K,$$

then this instruction will be modified:

1) if ab $\neq$ 0, then ab is replaced by ab – 1;

2) if ab = 0, then the third address 3777 of this instruction is changed to the beginning (entrance) of the corresponding (i.e., Nth) standard routine, while the standard routine itself is placed somewhere within the block β_N-β_K.

Let us assume that during operation of the main program, we reach instruction χ. If its third address 3777 was modified by the compiler to the address of the beginning of an appropriate standard routine, control is transferred to this standard routine. If the third address 3777 of instruction χ has not been changed, control will be transferred to location 3777. In this case, the interpreter comes into operation (naturally, it should first have been placed into the operating memory). It calls the appropriate standard routine into block γ-3777, changes the third address 3777 of instruction χ and transfers control to this standard routine. Upon repeated execution of the same instruction χ, control will be transferred either directly to the standard routine, or to the interpreter if the given standard routine has at this time been replaced in the block γ-3777 by any other standard routines.

Let us point out three more methods of using the compiler and interpreter (in addition to those discussed in §§ 3 and 4).

1. If the main program Π is divided into separate sections Π_0, Π_1, ..., Π_K executed one after the other in time, each being executed just once, then in working with the compiler it is possible to locate just those standard routines which are required for each of these sections alone in the operating memory. To do this, it is necessary to place instructions calling for the compiler at the beginning of each of the sections Π_i; the instructions should specifically indicate the appropriate block $\alpha_N^{(i)}$-$\alpha_K^{(i)}$. It is possible to indicate in every place precisely the same block α_N-α_K for the entire program Π, but then it is necessary to place the number $\underline{i}$ in digit positions ab of the instructions calling for the standard routines occurring in section $\underline{i}$.

2. It is possible to operate simultaneously with the compiling and interpreting programs. Let us assume, for example, that certain standard routines must be loaded from punched cards or from magnetic tape, while at the same time it is desirable for various reasons to use the interpreter. To do this we must:

a) place 00 in the ab digit positions of the instructions calling for those standard routines that must be executed with the compiler;

b) place any number not equal 00 into digit positions ab of the instructions calling for those standard routines that must be executed with the interpreter;

c) call for the compiler;

d) call for the interpreter, indicating for it a block γ-3777 that does not overlap the block β_N-β_K.

3. If the entire program cannot be placed into the operating

memory, and it is necessary to use a magnetic drum for its storage we may then: a) process with the aid of the compiler each portion of the main program and record on the magnetic drum not only that part of the main program, but also the corresponding blocks β_N-β_K; naturally, it is then necessary to call from the magnetic drum both that section of the main program, and the corresponding blocks with the standard routines; or else we can b) write a portion of the main program onto the magnetic drum in its initial form, and make use of the compiler each time the corresponding section is called into the operating memory.

§6. BLOCKING IN WORKING WITH THE COMPILING AND INTERPRETING PROGRAMS

The present system for using standard routines provides for monitoring the operation of several steps in the execution of the compiling and interpreting programs. This monitoring is not complete; in particular, the consequences of machine errors are not observed or eliminated (errors occurring in the arithmetic unit, the operating memory, or in the control unit). Only transfers of information (and nearly all of them) from external storage devices into the operating storage and the possibility of arranging the standard routines in the block set aside for them in the operating memory are monitored. When an error is found, the machine halts.

The reason for a stoppage may be determined from the contents of the instruction storage unit (register) (BZK) and the second-number order storage unit (BZ2P); there are light signals for these units on the control console.

Blocking in the Compiling Program

A call for the compiler, i.e., its transfer from the magnetic drum to the operating memory unit, is accompanied by the necessary modification of this program, carried out in four steps. The instructions calling for the compiler involve only the first auxiliary section of the program. If an error occurs during this call (for example, as a result of improper readout from the magnetic drum or owing to an error introduced by the programmer into the instructions calling for the compiler), the computer will halt. The address of the halt instruction (the contents of the TsUK register) is 0003. In the BZK, the instruction

33	-	-	0003

will be found.

When the machine is restarted, this section of the program will again be fed in. If desired, the second loading may be carried out not in the automatic mode, but by looping (i.e., execution of a single instruction).

The calls for the second and third portions of the compiler are not monitored (as in the case of the first section, they are placed into locations 0001-0017 of the operating storage). Proper transmission of the fourth, basic section of the program into the locations of the block A-A + 0157 is checked. If there is an error during this transfer, there will be a halt

33	0001	-	0003

The contents of the TsUK register (the address of the halt instruction) shall be omitted here and in the discussion to come, since the reason for the stoppage may be found even though the address is not available, while the address itself depends upon the particular locations in which the compiler is placed (the address depends on A). When the machine is restarted, all four sections of the compiler are loaded again.

The halt instruction

33	-	-	0001

may appear for various reasons that may be found from the contents of BZ2P. If BZ2P = 77, this means that the error was introduced by the standard routine. When the computer is restarted, this standard routine is loaded again. If the error was introduced from the punched cards, it is naturally first necessary to place the proper set of cards back in the card reader. The information may be reloaded either with the computer operating automatically, or by looping (it is necessary to execute from 4 to 10 instructions).

If BZ2P = 00, this means that the block β_N-β_K set aside for storage of the standard routines was too small. It is necessary to make an appropriate correction to the main program (in the instructions calling for the compiler); after this the entire problem must be run again (or that portion of the problem in which the erroneous call for the compiler is contained). If BZ2P = 22, this means that more than 22 standard routines were used in working with the compiler (including routines not mentioned in the main program but required for other standard routines). In this case, the compiler cannot be used (see §5).

Let us note that the number corresponding to the standard routine is sometimes omitted from the instructions calling for the standard routines. The machine will halt, and the instruction

33	3777	-	-

will be found in the BZK. If one of the instructions

31	-	~	0010

or

31	0001	~	0010

is found in the BZK, this indicates that the number N in the instruction calling for the standard routine corresponds to a routine that does not exist. In all cases, the first address of location 0014 will contain the location (i.e., the address) of the improper call for a standard routine.

A halt of the form

33	-	N	-

may appear in working with a standard routine having the number N. The reason for this is indicated in the description of the corresponding standard routine (or in the catalog).

<u>Blocking in the Interpreter</u>

When improper loading of the interpreting program occurs due either to improper readout from the magnetic drum or due to an error in the instructions calling for the interpreter, the computer will stop. The instruction

33	3400	-	-

will appear in the BZK.

When the machine is restarted in the automatic mode or with looping, the program is again loaded.

If an error occurs in reading a standard routine from the magnetic drum, the halt instruction

33	0001	-	0001

will appear, and the number 77 will appear in the BZ2P. When the machine is restarted in automatic operation or with looping, the standard routine will be reloaded. The same halt instruction

33	0001	-	0001

together with 00 in the BZ2P indicates that the longest of the standard routines used by the main program will not fit into the block set aside for the standard routines. The third address of location 3735 shows the location of the call for this standard routine, while the second address in location 3670 contains its number. In this case it is necessary to decrease γ in the instructions calling for the interpreting program.

If an instruction calling for a standard routine does not show the number of this routine, as in the case of the compiler, the halt instruction

33	3777	-	-

,

appears while if the number of a nonexistent standard routine is given, the machine will be halted by one of the instructions

31	-	~	3870

.

or

31	0001	~	3870
31	3777	~	3670

.

It is necessary to correct the improper instruction calling for the standard routine in the main program. The third address of location 3735 shows the location (address) of this instruction.

A halt instruction of the form

33	-	N	-

may appear when a standard routine with number N is being executed. The reason for this is shown in the description of the corresponding standard routine (or in the catalog).

A brief summary of the possible halts that may occur in the execution of the compiling and interpreting programs, with an indication of the criteria for these halts, their causes, and possible ways of correcting errors are given in Appendices 3 and 4.

§7. STANDARD ROUTINES

In this section we discuss the conditions that must be satisfied by standard routines if they are to be used with the compiling and interpreting programs.

1. When written, the lines of a standard routine have the floating address K-1, K+2, ..., $K + \lambda$, where λ is the length of the standard

routine. In coding, the letter K is replaced by 2000.

2. The entire standard routine is divided into two sections: the modified portion (lines K + 1, K + 2, ..., K + λ_P, where λ_P is the length of the modified portion) and the unmodified portion (lines K + + λ_P + 1, ..., K + λ). When the standard routine is stored in the operating memory, all of the internal addresses in the modified section will be transformed appropriately. In this case, any address B $\geq$ $\geq$ 2000 is considered to be an internal address of the form K + (B – – 2000) and it will be transformed appropriately (for one exception, see Subsection 5). When the standard routines are stored in the memory, the lines of the unmodified portion do not change (for one exception, see Subsection 5).

3. The instruction K + 1 is the entrance to the standard routine. It should be so written that the main program calls for it by the instruction

$$\boxed{77}\boxed{abcd}\boxed{efgh}\boxed{K+1},$$

which is executed in the TsUK.

Here the octal digits $\underline{a}$, $\underline{b}$ and $\underline{e}$, $\underline{f}$, $\underline{g}$ and $\underline{h}$ may have arbitrary values (they are required for the compiling and interpreting programs). Additional information may be placed into digit positions $\underline{c}$, $\underline{d}$, as well as in succeeding lines.

4. It is desirable for all positions in the operating memory to be available for execution of the standard routine (after K has been replaced by the appropriate number).

The limitation

$$K \geq 0020,$$

is permissible; stronger limitations are extremely undesirable, and in any case they can only take the form

$$K \geq K_0.$$

5. Calls for various standard routines within a given standard routine should be carried out in accordance with the general rules for a standard-routine call (see Subsection 3). If it is necessary to go to the instruction numbered N, we write the instruction

$$\begin{array}{|c|c|c|c|}\hline 77 & 00cd & N & 3777 \\ \hline\end{array}.$$

There should be no lines of the form

$$\begin{array}{|c|c|c|c|}\hline 77 & \sim & \sim & 3777 \\ \hline\end{array}.$$

in either the modified or unmodified sections that do not call for a standard routine. In all lines of this type, the third address will be changed to the beginning of the appropriate routine. In addition to this change and the transformation described in Subsection 2, there are no changes in the standard routine when it is placed into the operating storage.

6. Wherever (as permitted, see Subsection 4) a standard program has not been placed into the operating memory, it should contain no new lines of the form

$$\begin{array}{|c|c|c|c|}\hline 77 & \sim & \sim & 3777 \\ \hline\end{array}.$$

except for those mentioned in Subsection 5. In other words, in a standard routine, control should not be transferred to the last line of this routine

$$\begin{array}{|c|c|c|c|}\hline 77 & \sim & \sim & I+\lambda \\ \hline\end{array}.$$

7. If the standard routine provides for any type of blocking, the blocking instructions should take the form

$$\begin{array}{|c|c|c|c|}\hline 33 & - & N & - \\ \hline\end{array},$$

where N is the number of the given standard routine. When halted in this fashion, the computer should depart from the given standard routine when restarted.

8. For a standard routine to operate with the interpreter, at each call for this standard routine (for the case in which there may

be several types of call) all information needed to load the standard routine and place it into operation should be given.

§8. THE LIBRARY OF STANDARD ROUTINES

In this section we discuss the basic rules for organizing a library of standard routines to be used with the compiling or interpreting programs.

Two additional lines are placed before each standard routine. The second indicates the length of the modified section in the form

$$\boxed{-}\boxed{-}\boxed{-}\boxed{\lambda_n},$$

and the first gives the complement to the number

$$\boxed{77}\boxed{3777}\boxed{3777}\boxed{3777}$$

of the check sum of the standard routine and the previously-given second additional line (the summation is carried out in accordance with the "cyclic addition" operation No. 60). Henceforth when we use the term standard program we shall always mean the standard program together with these two additional lines.

It is desirable to store standard routines with low numbers N, for example, $0001 \leq N \leq 0077$, on magnetic drum or magnetic tape. On a zero-type magnetic drum, an appropriate number of locations with addresses from 2.2001 and above are set aside for the table of standard routines.

This table contains information on those standard routines that are written on the magnetic drum or magnetic tape: for standard routine number N, location 2.2000 + N must contain the first instruction of the pair of instructions used to load this standard routine into the operating memory.

On each magnetic drum, a portion of a block with addresses from 2.2000 to 2.3777 is set aside for standard routines. On magnetic tape, each standard routine should occupy one zone. The zone numbers should run from 1 to 63 (77 in the octal system). The zones should be arranged in increasing order (of their numbers).

If the standard routine is stored on punched cards, one additional punched card should be placed before the card pack; two lines are punched in it: the first is of the same form as is used in the table of standard routines (with an allowance for loading from punched cards), while the second indicates the number of the given standard routine:

30	0100	-	$\lambda+1$
-	-	N	-

.

Here λ is the length of the standard routine alone (with no allowance for the two additional lines mentioned in the beginning of this section); N is the number of the standard routine.

§9. FLOW CHART AND DESCRIPTION OF COMPILING PROGRAM

The compiling program is loaded into the operating memory and placed into operation in four steps. The first auxiliary portion of the compiler is executed at locations 0001-0017. Here the correctness of the call for the compiler is checked, certain constants are prepared, and the second auxiliary portion is called. The latter (it occupies locations 0006-0017) calls the main portion of the compiler, makes certain corrections in it, and calls the third auxiliary portion of the compiler, which concludes preliminary processing of the main portion of the compiler and transfers control to it; the third portion occupies locations 0006-0016.

The main portion of the compiler must be modified so that it can be executed at locations A-A + 0157 set aside for it. The auxiliary sections execute some of these modifications; the remaining modifications are carried out by elements of the main portion, just as they modify standard routines. Below we give the flow charts and a description of the compiler (more accurately, its basic portion), giving no consideration to the preliminary preparation, other than what we have just mentioned. We shall give in the same form a description with comments of the compiler, using floating addresses. In addition, we shall give the final code for the entire compiling program.

Description of Compiler Operators

The entrance to the program is operator 37.

Operator 1 – formation of constants for checking conclusion of standard-routine modification cycle.

Operators 2, 3 and 4 – calculation of correction Δ for transformation of internal addresses; if $K \geq 2000$, then $\Delta = K - 2000$, if $K <$ < 2000, then $\Delta = K$ and the clearance instruction for ε_1 is sent (equal to 2000 at address A_1).

Operator 5 – reading of instructions from standard routine and indexing of material read.

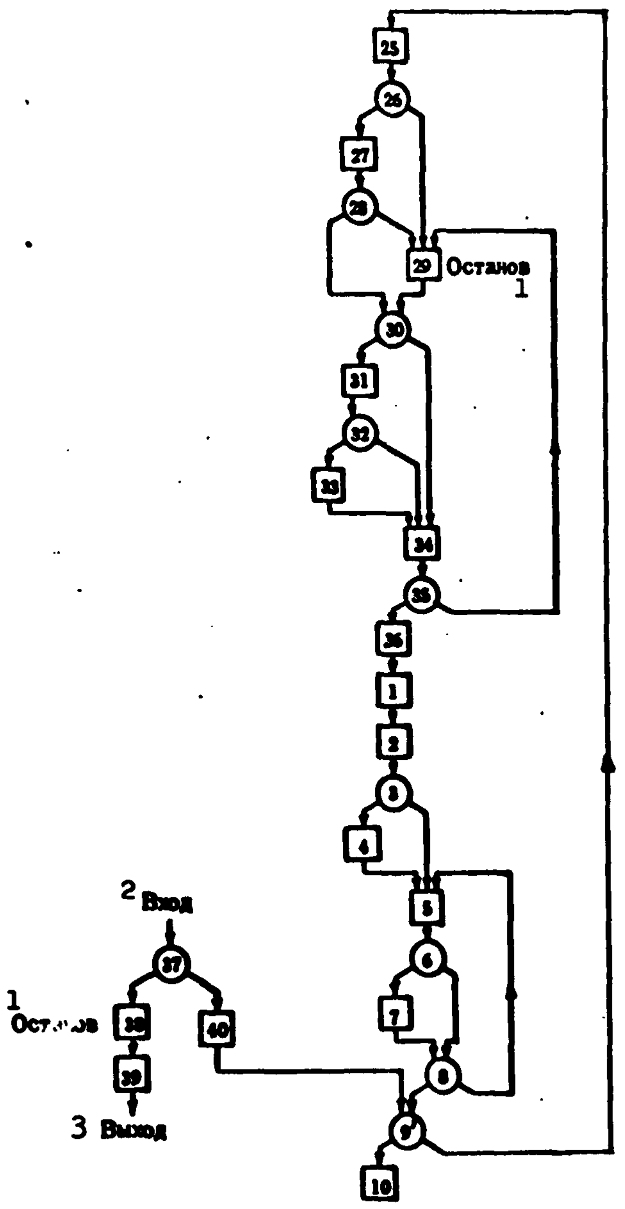
Operator 6 – jump to operator 8 if the instruction selected is a call for a standard routine.

Operator 7 – transformation of selected instruction (substituting $K + i$ for addresses of the form $2000 + i$) and placing it in standard routine.

Operator 8 – check to see whether processing of standard routine has been concluded; if this is not the case, jump to operator 5, otherwise jump to operator 9.

Operator 9 = 9' (prior to execution of operator 10) – check to see whether all standard routines have been loaded from punched cards; if they all have been loaded, jump to operator 10, if not, jump to operator 25.

Operator 9 = 9" (following execution of operator 10) – transfer con-



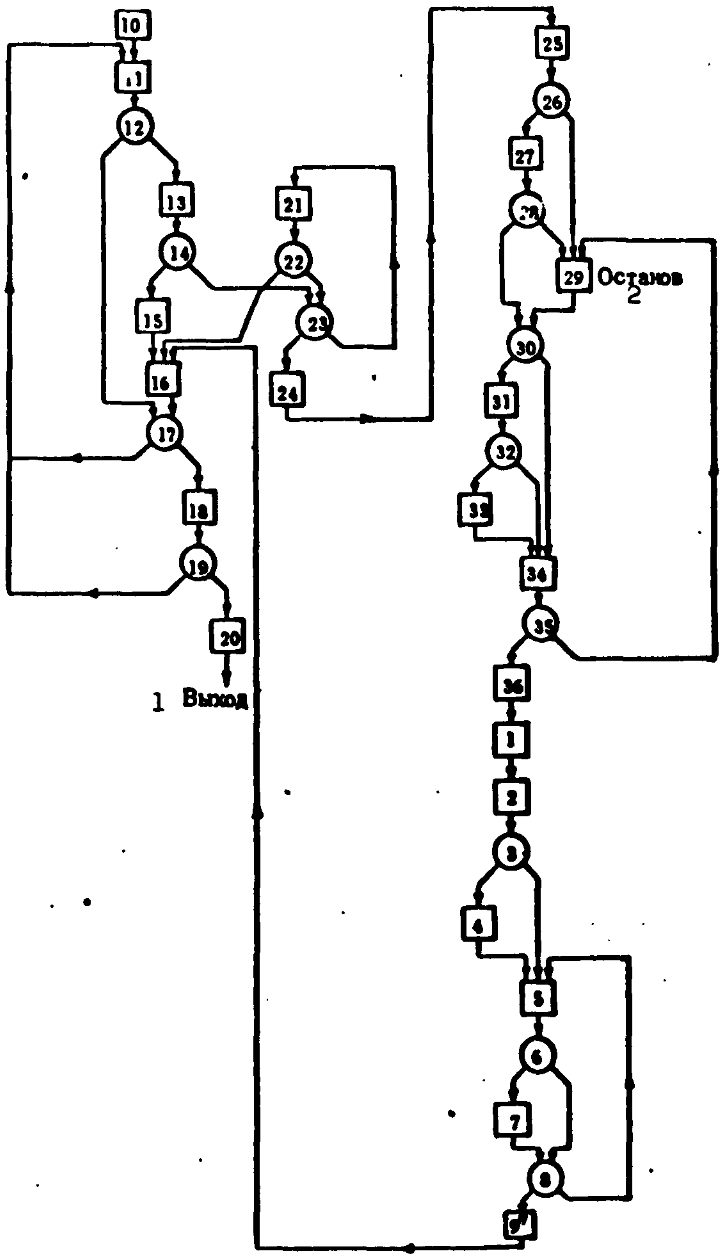
Flow chart for the portion of the compiler preceding operator 10. 1) Halt; 2) entrance; 3) exit.

trol to operator 16.

Operator 10 – conversion of operator 9' into 9".

Operator 11 – reading lines from program (from block α_N-α_K).

Operator 12 – check to see whether selected line is a call for a standard routine; if it is, code to operator 13, otherwise jump to



Flow chart for portion of compiling program following execution of operator 10. 1) Exit; 2) halt.

operator 17.

Operator 13 – execution of prepared operation, including preparation of selected material from table of initial location.

Operator 14 – jump to operator 23 if ab = 0, or to operator 15 if ab $\neq$ 0.

Operator 15 – decrease ab by 1: ab – 1 = >ab.

Operator 16 – place the altered line back in the program.

Operator 17 – indexing; check to see whether the entire block α_N-α_K has been examined; if it has, go to operator 18, if not, jump to operator 11.

Operator 18 – preparation for examination of new block $\alpha_N^!$-$\alpha_K^!$.

Operator 19 – check to see whether the new block has been examined; if it has, jump to operator 11; if it has not, go to operator 20.

Operator 20 – exit from compiler.

Operator 21 – read from table of initial locations and pick out number of standard routine.

Operator 22 – compare numbers of standard routines; if numbers agree, jump to operator 16; if not, go to operator 23.

Operator 23 – check to see whether the end of the table of initial location has been reached; if it has been reached, go to operator 24; if not, jump to operator 21.

Operator 24 – formation of instruction for reading from table of standard routines located on magnetic drum.

Operator 25 – load one line from table of standard routines or two lines from extra punched card, form λ, K.

Operator 26 – check to see whether there is a location for one line in the table of initial locations (a block of 22 locations is set aside for it); if there is, go to operator 27; if not then halt (operator 29).

Operator 27 – place line of form

-	-	I	$I+1$

into table of initial locations, indexing of material placed, formation of instruction for loading standard routine.

Operator 28 – check to see whether there is an adequate free area in the block for the given standard routine; if the area is sufficient, jump to operator 30; if not then go to operator 29.

Operator 29 – halt.

Operator 30 – check to see whether the standard routine is located on magnetic tape; if it is, go to operator 31; if not, go to operator 34.

Operator 31 – store new number of zone on magnetic tape.

Operators 32 and 33 – rewind magnetic tape where necessary.

Operator 34 – load standard routine into operating memory.

Operator 35 – check to see whether standard routine has been properly loaded; if it has, go to operator 36; if not, jump to operator 29.

Operator 36 – preparation of variable instructions for operators 1, 5 and 7 that modify the standard routine, transfer control to operator 1.

Operator 37 – entrance to compiling program, check to see whether compiling program has been properly introduced; if it has, jump to operator 40; if not, go to operator 38.

Operator 38 – halt.

Operator 39 – preparation for repeated entrance of compiler and transfer of command to it.

Operator 40 – correction of certain instructions and preparation of the content of certain working locations (s1, s2, ..., s6) needed for execution of the compiler. Transfer control to operator 9.

The general sequence of execution of the compiler is as follows: after preliminary "adjustment" carried out by operator 40, the standard routines are loaded one after the other from punched cards. For each new standard routine placed into the block $\beta_N + \beta_K$, one line of the form

$$\boxed{-}\boxed{-}\boxed{N}\boxed{K+1}.$$

is taken from the table of initial locations, where N is the number of the given standard routine; K + 1 is its initial location. The table of initial locations is stored at the location of operators 37-40. It should be no more than 22 lines long (in the decimal system). When it is loaded from the punched cards, the length of the standard routine and its number are taken by operator 25 from the two lines punched in the extra punched card (see §8). Modification of the standard routine is carried out by operators 5, 6, 7 and 8. Each address required in the modification is changed as follows: if $K \geq 2000$, then $\Delta = K-2000$ is added to the address; where $K < 2000$, the 11th digit in the address is cleared, i.e., 2000 is subtracted, and then K is added. After the loading of standard routines from punched cards has been concluded, inspection of the program begins, or more accurately, inspection of the block $\alpha_N - \alpha_K$ (operators 11 through 17). If in a line of the form

$$\boxed{77}\boxed{abcd}\boxed{N}\boxed{3777}$$

$ab \neq 0$, then ab is replaced by ab – 1. If, however, ab = 0, a search in the table of initial positions is carried out (by operators 21, 22, and 23) for a line with the numbers N. If there is such a line, the third address 3777 is changed by the appropriate K + 1; if there is no such line in the table of initial locations, the standard routine of number N is placed into the operating memory in block $\beta_N + \beta_K$. Information on where this program has been stored and its length is obtained from the table of standard routines (operator 25 takes the Nth line of this table; the Nth line contains information on standard routine number N – see §8). In one of the working locations (s3) there is stored the number of the last zone containing standard programs read from magnetic tape. At first there is a zero in this location, and the tape is rewound to the beginning. If the number of the next zone does not exceed the number of the last zone read, the magnetic tape is wound back (operator 33). After the entire block α_N-α_K has been inspected, all standard routines called are inspected (operator 18 changes α and β appropriately, and after this, control is transferred to operator 11). This is done so as to allow for the possibility of a call within one standard routine for other standard routines. If new standard routines have been called into the operating memory, it is necessary to go through still another inspection, etc. If as a result of the inspection there has been no call (determined by operator 19), operation of the compiler has been concluded (operator 20).

Адрес 1	Команда 2					№ оператора 3	4 Примечания
1	2				3	4	5
A	22	A+15	K	r1	●	1	5 Образование константы для проверки окончания цикла
A+1	62	в1	B+7	r2		2	K – 2000
2	76	r2	B+13	A+5		3	6 Если $K \geq 2000$, то – переход к оператору 5
3	62	в1	A+71	r2		4	7 K (для $K < 2000$)
4	45	A+55	B+7	A+12			8 Засылка команды гашения
5	16	–	K+1	r3	●	5	9 Выборка команды из СП
6	22	A+5	B+6	A+5			10 Переадресация выборки
7	17	r3	B+14	r4		6	11 Выяснение, является ли выбранная команда обращением к СП. Если да, то – переход к оператору 8
10	35	r4	B+14	A+16			
1	17	r3	B+10	r4		7	12 Выделение $e_1 e_2 e_3$
2	20	r3	r4	r8	●		13 Гашение $e_1 e_2 e_3$, если $K < 2000$
3	26	r4	0112	r4			
4	51	r4	r2	r4			
5	22	r3	r4	K+1	●		14 Засылка в СП измененной команды
6	22	A+15	A+71	A+15		8	15 Переадресация засылки
7	75	A+15	r1	A+5			16 Если обработка СП не закончена, то – переход к оператору 5
20	62	в2	B+5	в2	●	9	17 Затем $\boxed{35}\boxed{-}\boxed{-}\boxed{A+36}$
1	36	–	в2	A+64			18 Если введены не все СП с перфокарт, то – переход к оператору 25
2	17	A+61	B+14	A+20		10	19 Изменение оператора 9
3	26	A+36	0025	r5		11	20 Образование команды

21 Продолжение программы

1	2				3	4	5
A+24	22	r5	A+71	A+25		11	22 Выборки строки из программы
5	16	$e_н$	–	0001	●		23 Выборка из программы
6	17	0001	B+14	r6		12	24 Выяснение, является ли выбранная строка обращением к СП. Если нет, то – переход к оператору 17
7	75	r6	B+14	A+37			
30	20	0001	B+12	r7		13	$\boxed{77}\boxed{A_1}\boxed{A_2}\boxed{-}$
1	17	0001	B+13	r10			$\boxed{-}\boxed{-}\boxed{N}\boxed{-}$
2	17	0001	B+11	r11			$\boxed{-}\boxed{ab00}\boxed{-}\boxed{-}$
3	65	A+101	0065	A+46			25 Засылка команды выборки из ТН
4	35	r11	–	A+52		14	26 Если $ab = 0$, то – переход к оператору 23
5	22	r7	B+11	r7		15	$ab - 1 \Rightarrow ab$
6	16	r7	r6	$e_н$	●	16	27 Засылка строки в программу
7	22	A+36	A+71	A+36		17	28 Переадресация: +1 (III)
40	75	A+36	в5	A+23			29 Выяснение, просмотрена ли вся программа. Если нет, то – переход к оператору 11
1	22	B+3	в1	A+36		18	30 Подготовка к новому просмотру
2	22	B+3	в6	в5			31 Засылка новых $e_н$, $e_к$ и $\beta_к$
3	–	в1	–	в6			
4	75	A+36	в5	A+23		19	32 Если нужен новый просмотр, то – переход к оператору 11
5	74	–	–	L+5		20	33 Выход из компилирующей программы
6	16	–	ТН$_н$	r6	●	21	34 Выборка из ТН $\boxed{-}\boxed{-}\boxed{N}\boxed{K+1}$
7	22	A+46	B+6	A+46			35 Переадресация выборки

Compiler With Floating Addresses (continued)

1	2				3	4	5
A+50	17	r6	B+13	r12			$\boxed{-}\boxed{-}\boxed{N}\boxed{-}$
1	35	r12	r10	A+36		22	36 Если номера СП совпали, то – переход к оператору 16
2	75	A+46	B+5	A+46		23	37 Выяснение, кончилась ли ТН. Если нет, то – переход к оператору 21
3	22	B+17	r10	A+54		24	38 Образование команды выборки из ТСП
4	30	0100	–	0001	•	25	39 Затем $\boxed{30}\boxed{0402}\boxed{2000+N}\boxed{-}$
5	31	r3	r4	r3			40 Выборка из ТСП (r4+1=r10)
6	17	r4	B+12	r13			$\boxed{-}\boxed{-}\boxed{-}\boxed{\lambda+1}$
7	62	s1	r13	r14			$\boxed{-}\boxed{-}\boxed{-}\boxed{K}$
60	22	r14	A+71	s1			$\boxed{-}\boxed{-}\boxed{-}\boxed{K+1}$
1	–	s1	–	r6			
2	35	A+63	B+4	A+71		26	41 Если нехватает места для ТН, то останов – 29
3	22	r10	r6	$ТН_н$	•	27	42 Затем $ТН_к + 1$; Засылка $\boxed{-}\boxed{-}\boxed{N}\boxed{K+1}$
4	22	A+63	A+71	A+63			43 Переадресация засылки
5	22	B+5	B+6	B+5			44 Увеличение $ТН_к$ на 1 $\boxed{-}\boxed{-}\boxed{K}\boxed{-}$
6	26	r14	0013	r15			
7	22	B+16	r15	A+103			45 Образование команды считывания СП
70	36	s4	r14	A+72		28	46 Если СП умещается в памяти, то – переход к оператору 30, иначе – останов
1	33	–	–	0001		29	
2	17	r4	A+110	r16		30	47 Если СП не находится на МЛ, то – переход к оператору 34
3	36	r16	B+6	A+102			
4	–	s3	–	r16		31	48 Запоминает новый номер зоны на МЛ
5	17	r4	B+12	s3			

1	2				3	4	5
A+76	36	r16	s3	A+102		32	49 Если обратная перемотка МЛ не нужна, то – переход к оператору 34
7	15	r4	B+10	A+100		33	50 Обратная перемотка МЛ
100	30	$2000+\varepsilon_{сп}$	$K_л$	~	•		
1	31	–	$ТН_н$	r6			
2	30	$\varepsilon_{сп}$	$A_{сп}$	λ+1	•	34	51 Ввод СП в оперативное запоминающее устройство
3	31	0001	K–1	r16	•		
4	75	r16	B+15	A+71		35	52 Если СП введась неверно, то – переход к оператору 29
5	22	B	r15	A+5		36	53 Подготовка переменных команд для обработки СП
6	22	B+1	s1	A+15			
7	22	B+2	r15	A			
110	34	0200	–	A			
1	35	0005	B+15	A+115		37	
2	33	0001	–	0003		38	
3	62	0001	A+112	A+114		39	
4	34	~	–	L+1			
5	–	c1	–	A+20		40	
6	–	c2	–	A+55			
7	26	0001	0026	0010			$\boxed{-}\boxed{L+4}\boxed{-}\boxed{-}$
120	22	A+121	0010	A+121			
1	26	L+4	0113	0005			$\boxed{-}\boxed{n}\boxed{\rho_н}\boxed{\beta_к}$
2	17	0005	B+12	0010			$\boxed{-}\boxed{-}\boxed{-}\boxed{\rho_к}$
3	22	0010	A+71	s6			$\boxed{-}\boxed{-}\boxed{-}\boxed{\beta_к+1}$
4	26	0005	0113	s2			$\boxed{-}\boxed{-}\boxed{n}\boxed{\rho_н}$
5	17	s2	B+12	s4			$\boxed{-}\boxed{-}\boxed{-}\boxed{\beta_н}$
6	–	s6	–	s1			
7	62	A+121	A+112	A+130			
130	26	L+3	0113	0002			$\boxed{-}\boxed{77}\boxed{\alpha_н}\boxed{\alpha_к}$
1	17	0002	B+12	0010			$\boxed{-}\boxed{-}\boxed{-}\boxed{\alpha_к}$
2	22	c3	0010	s5			
3	26	0002	0113	0010			$\boxed{-}\boxed{-}\boxed{77}\boxed{\alpha_н}$

Compiler With Floating Addresses (continued)

1	2				3	4	5
A+134	20	c4	0010	A+36			
5	22	0001	A+71	A+45			54 Передача управления на оператор 9
6	74	–	в3	A+20			

1) Address; 2) instruction; 3) number of operator; 4) comments; 5) formation of constants for checking conclusion of cycle; 6) if $K \geq 2000$, jump to operator 5; 7) K (for $K < 2000$); 8) enter clear instruction; 9) take instruction from standard routine; 10) indexing of material selected; 11) check to see whether instruction selected calls for standard routine. If it does, jump to operator 8; 12) pick out $\varepsilon_1\varepsilon_2\varepsilon_3$; 13) clear $\varepsilon_1\varepsilon_2\varepsilon_3$, if $K < 2000$; 14) place changed instruction into standard routine; 15) indexing of instruction placed; 16) if processing of standard routine has not been concluded, jump to operator 5; 17) then; 18) if not all of the standard routines have been loaded from punch cards, jump to operator 25; 19) change operator 9; 20) form instructions; 21) continuation of program; 22) take lines from program; 23) read from program; 24) see whether line selected calls for standard routine; if it does not, jump to operator 17; 25) transmit instruction to read from TN; 26) if ab = 0, jump to operator 23; 27) place lines into program; 28) indexing: +1 (III); 29) see whether entire program has been inspected. If it has not, jump to operator 11; 30) preparation for new inspection; 31) transmit new α_N, α_K and β_K; 32) if a new inspection is necessary, jump to operator 11; 33) exit from compiler; 34) read from TN; 35) indexing of material read; 36) if standard-routine numbers agree, jump to operator 16; 37) see whether TN is concluded; if not, jump to operator 21; 38) formation of instructions to read from TSP; 39) then; 40) read from TSP; 41) if area available for TN is insufficient, halt – 29; 42) then $TN_K + 1$; transmit; 43) index material sent; 44) increase TN_K by 1; 45) form instructions for reading SP; 46) if SP will fit into memory, jump to operator 30, otherwise halt; 47) if SP is not on ML, jump to operator 34; 48) store new number of zone on ML; 49) if it is not necessary to rewind ML, jump to operator 34; 50) rewind ML; 51) load SP into operating memory; 52) if SP is improperly introduced, jump to operator 29; 53) prepare variable instructions for processing SP; 54) transfer control to operator 9.

Notes: *) variable instructions; SP) standard routine; TN) table of initial location; TN_N) beginning of table of initial positions; TN_K) end of table of initial locations; TSP) table of standard routines; ML) magnetic tapes.

Constants for Compiler

B	16	-	0001	$r3$
B+1	22	$r3$	$r4$	-
2	22	A+15	-	$r1$
3	16	$r7$	$r6$	-
4	22	$r10$	$r6$	$TH_{KM}+1$
5	16	-	TH_K+1	$r6$
6	30	-	0001	-
7	67	-	-	2001
10	-	2000	2000	2000
1	-	3700	-	-
2	-	-	-	3777
3	-	-	3777	-
4	77	-	-	3777
5	77	3777	3777	3777
6	31	-	3777	$r16$
7	30	0402	2000	-

Note: TN_{KM}) end of block set aside for table of initial locations; TN_K) end of table of initial locations.

Content of Certain Working Locations in Compiler

s_1	-	-	-	$I+1$
s_2	-	-	n	β_n
s_3	-	-	N_1	-
s_4	-	-	-	β_n
s_5	16	r_7	r_6	α_K+1
s_6	-	-	-	β_K+1

Comment: N_1 is the number of the magnetic-tape zone.

Certain Additional Constants for Blocks 37-40 of the Compiler

c_1	62	s_2	$B+6$	s_2
c_2	31	r_3	r_4	r_3
c_3	16	r_7	r_6	0001
c_4	16	r_7	0061	–

Absolute and Relative Addresses Assigned to Constants and Working locations in Compiler

r1	A
r2	A + 25
r3	A + 46
r4	A + 102
r5	0014
r6	0016
r7	0015
r10	A + 103
r11	0017
r12	0001
r13	0012
r14	0013
r15	A + 12
r16	0010
s1	A + 157
s2	0003
s3	0002
s4	0004
s5	0005
s6	0007

B	A + 137
B + 1	A + 140
B + 2	A + 141
B + 3	A + 142
B + 4	A + 143
B + 5	A + 144
B + 6	A + 145
B + 7	A + 146
B + 10	A + 147
B + 11	A + 150
B + 12	A + 151
B + 13	A + 152
B + 14	A + 153
B + 15	A + 154
B + 16	A + 155
B + 17	A + 156
c1	A + 114
c2	A + 130
c3	A + 45
c4	A + 36

COMPILING PROGRAM

First Auxiliary Portion. Occupies Locations 2.3001-2.3017 on Magnetic Drum (NO)

1	34	-	0001	0002
2	35	0005	0017	0006
3	33	-	-	0003
4	62	0001	0003	0005
5	11	3342	0621	3707
6	26	0001	0026	0002
7	22	0010	0002	0010
10	17	-	0017	0002
11	26	0002	0013	0005
12	26	0005	0013	0003
13	16	0002	0003	0003
14	16	0003	0005	0004
15	30	0402	3020	0011
16	31	-	0006	-
17	-	-	-	3777

Second Auxiliary Portion. Occupies Locations 2.3020-2.3031 on Magnetic Drum (NO)

6	22	0007	0002	0007
7	-	0002	1612	0157
10	22	0012	0005	0012
11	30	0402	3043	0156
12	31	2554	-	0005
13	22	0014	0003	0014
14	22	0055	0004	0055
15	30	0402	3032	0010
16	31	-	0006	-
17	34	-	-	0006

Third Auxiliary Portion. Occupies Locations 2.3032-2.3042 on Magnetic Drum (NO)

6	20	0007	0003	0007
7	22	-	0004	-
10	22	0007	0016	0007
11.	65	0002	0001	0002
12	76	0002	0006	0007
13	22	0014	0003	0014
14	22	0025	0003	0013
15	22	0016	0002	0016
16	34	0001	-	0001

Main Portion. Occupies Locations 2.3043-2.3221 on Magnetic Drum (NO)

A	22	0046	0102	0145
A+1	62	0157	0146	0025
2	76	0025	0152	0005
3	62	0157	0071	0025
4	45	0055	0146	0012
5	16	0100	0021	0046
6	22	0005	0145	0005
7	17	0046	0153	0102
10	35	0102	0153	0016
1	17	0046	0147	0102
2	-	-	-	-
3*	10	2370	1600	0114
4	51	0102	0025	0102
5	22	0046	0102	0021
6	22	0015	0071	0015
7	75	0015	-	0005
20	34	-	-	0111
1	36	-	0003	2055
2	17	2052	2154	2021
3	26	2037	0026	0014

A+24	22	0014	2072	2026
5	26	0102	0112	0102
6	17	0001	2154	0016
7	75	0016	2154	2040
30	20	0001	2152	0015
1	17	0001	2153	2104
2	17	0001	2151	0017
3	65	2102	0065	2047
4	35	0017	-	2053
5	22	0015	2151	0015
6	16	0015	0061	-
7	22	2037	2072	2037
40	75	2037	0005	2024
1	22	2143	2160	2037
2	22	.2143	0007	0005
3	-	2160	-	0007
4	75	2037	0005	2024
5	16	0015	0016	0001
6	-	-	-	-
7	22	2047	2146	2047
50	17	0016	2153	0001
1	35	0001	2104	2037
2	75	2047	2145	2047
3	22	2157	2104	2055
4	30	0100	-	0001
5	31	0046	0102	0046
6	17	2103	2152	0012
7	62	2160	0012	0013
60	22	0013	2072	2160
1	-	2160	-	0016
2	35	2064	2144	2072
3	22	2104	0016	2112
4	22	2064	2072	2064
5	22	2145	2146	2145
6	26	0013	0013	2013
7	22	2156	2013	2104
70	36	0004	0013	2073
1	33	-	-	0001
2	17	2103	2111	0010

Main Portion. Occupies Locations 2.3043-2.3221 on Magnetic Drum (NO) (continued)

A+ 73	36	0010	2146	2103
4	-	0002	-	0010
5	17	2103	2153	0002
6	36	0010	0002	2103
7	15	2103	2150	2101
100	-	-	-	-
1	31	-	2112	0016
2	-	-	-	-
3	-	-	-	-
4	75	0010	2155	2072
5	22	2140	2013	2006
6	22	2141	2160	2016
7	22	2142	2013	2001
110	34	0200	-	2001
1	35	0005	2155	2116
2	33	0001	-	0003
3	62	0001	2113	2115
4	62	0003	2146	0003
5	-	2115	-	2021
6	-	2131	-	2056
7	26	0001	0026	0010
120	22	2122	0010	2122
1	26	-	0113	0005
2	17	0005	2152	0010
3	22	0010	2072	0007
4	26	0005	0113	0003
5	17	0003	2152	0004
6	-	0007	-	2160
7	62	2122	2113	2131
130	31	2047	2103	2047
1	17	0002	2152	0010
2	22	2046	0010	0005
3	26	0002	0113	0010
4	20	2037	0010	2037
5	22	0001	2072	2046
6	74	-	0002	2021
7	16	-	0001	2047
140	22	2047	2103	-
1	22	2016	-	2001
2	16	0105	0016	-
3	22	2104	0016	2140
4	16	-	2112	0016
5	30	-	0001	-
6	67	-	-	2001
7	-	2000	2000	2000
150	-	3700	-	-
1	-	-	-	3777
2	-	-	3777	-
3	77	-	-	3777
4	77	3777	3777	3777
5	31	-	3777	0010
6	30	0402	2000	-

* All lines are complements of the check sum.

§10. FLOW CHART AND DESCRIPTION OF INTERPRETING PROGRAM

When the interpreter is loaded into the operating memory, only operators 1 to 7 are executed. They check that the interpreter has been read properly from the magnetic drum (if an error occurs during readout, the machine stops, and readout is then repeated), and prepares it for execution, introducing into the appropriate location the beginning γ of the block γ-3777 set aside for standard routines and the interpreter. The interpreter itself (without operators 1-7, which are now no longer needed) occupies locations 3665-3777. At the beginning of the block γ-3664 is located the table of control transfers; the length of this table increases during execution of the main program, and equals the number of various standard routines that have already participated in the run. The remaining portion of the block γ-3664 is set aside for standard routines.

If in the execution of the main program a call for a standard routine

$$\chi \quad \begin{array}{|c|c|c|c|}\hline 77 & \sim & N & 3777 \\ \hline\end{array}.$$

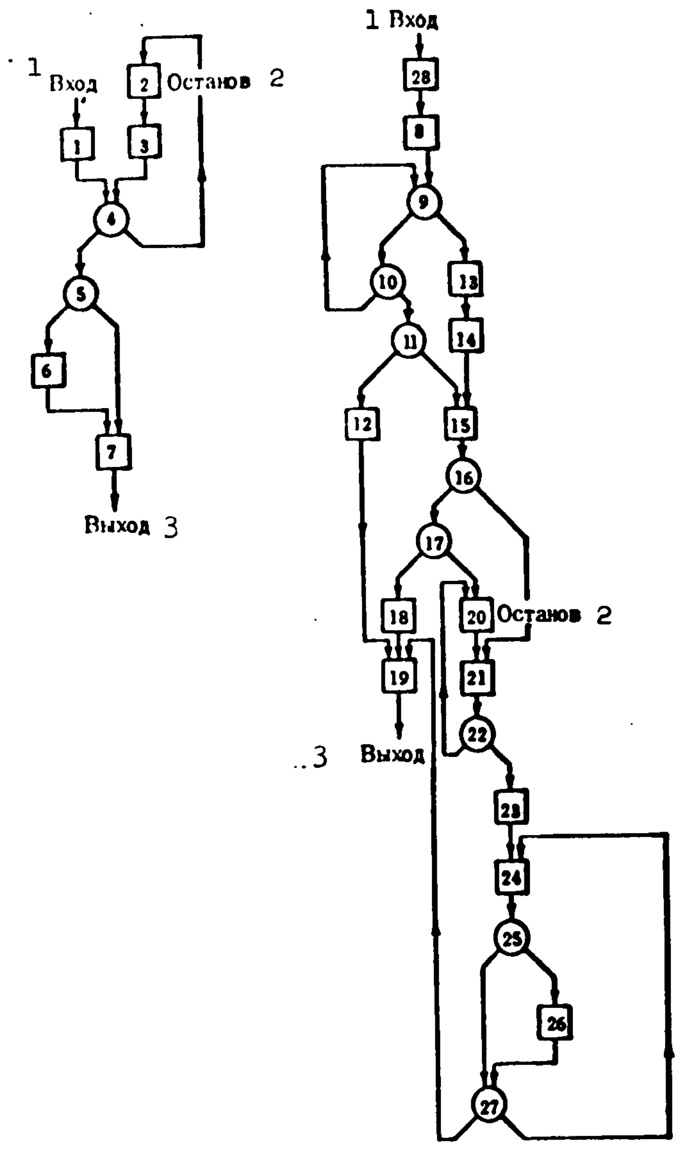
is encountered, it is necessary to transfer control to the interpreter (operator 28). The latter first of all inspects the table of control transfers.

If the given standard routine (with number N) has not yet participated in the run, the table of control transfers will not have a line of the required form. In this case, the Nth standard routine is entered into a free portion of the block γ-3664; its first location will occur at K + 1. This standard routine, naturally, will be appropriately modified. In the table of control transfers a line of the form

$$\gamma + i \quad \begin{array}{|c|c|c|c|}\hline 77 & - & N & K+1 \\ \hline\end{array}.$$

will be entered, and in instruction χ, the third address will be

Flow Chart for Interpreter



1) Entrance; 2) halt; 3) exit.

changed by $\gamma + 1$. After this, the interpreter concludes its operation and transfers control to location χ.

If in inspecting the table of control transfers, a line of the type mentioned is found containing the given number in the second address, the standard routine will not be entered into the operating memory, and in instruction χ, the third address 3777 will be replaced by the address $\gamma + 1$ of the line found. After this, control will again be transferred to instruction χ.

If the standard routines called take up a considerable portion of the block γ-3664, and the available space proves insufficient for a new standard routine, then all of block γ-3664 not occupied by the table of control transfers will be freed; in the table of control transfers, all transfers of control to older standard routines are "annulled" (the third address is changed to 3777), and the new standard routine is placed where the old was stored.

If during further execution of the main program calls are again encountered for "old" (already erased) standard routines, these routines will be loaded into the free portion of the block γ-3664, and in the appropriate line of the control-transfer table, the third address 3777 will be changed to the entrance to the standard routine.

If the entire section of block γ-3664 not occupied by the table of control transfers turns out to be insufficient to store a single standard routine, the machine will halt.

It should be noted that during execution of the interpreter, not even a single working location outside of this program is used.

Description of Interpreter Operators

The interpreter has two entrances; operator 1 – for calling the interpreter and preparing it for operation, and operator 28 – for transferring control to the interpreter at the instant that any standard routine is called for.

Operator 1 (entrance to interpreter when it is prepared for execution) – transfer of control to operator 4 with a jump to central control.

Operator 2 – halt when the interpreter is improperly loaded.

Operator 3 – repeated entrance of the interpreter.

Operator 4 – check to see whether interpreter has been correctly introduced; if not, jump to operator 2; if it has been correctly loaded, go to operator 5.

Operators 5 and 6 – take γ from the L + 3rd line; if the value of γ does not exceed 2000, γ is taken equal to 3400.

Operator 7 – prepare interpreter for execution (place γ into appropriate positions in several working locations) and return to main program.

Operator 8 – form instructions for exiting from interpreter (operator 19); take number N of standard routine and prepare for inspection of control-transfer table.

Operator 9 – check to see whether table of control transfers has been inspected; if it has, jump to operator 13; if not, go to operator 10.

Operator 10 – read from table of control transfers and compare corresponding N' with given N; if $N' = N$, go to operator 11; if $N' \neq N$, jump to operator 9.

Operator 11 – check to see whether the standard routine with number N has been placed into the operating memory (more accurately, check to see that it has not been replaced by another standard routine); if the standard routine is present in its entirety, go to operator 12; if not, jump to operator 15.

Operator 12 – execute (at MUK) operator 14, and then transfer control to exit (operator 19).

Operator 13 — increase length of control-transfer table by 1.

Operator 14 — change third address (3777) in instruction χ calling for standard routine, and change it into instruction transferring control to line in control-transfer table corresponding to the given standard routine.

Operator 15 — read corresponding line from standard-routine table, form λ, K, transmit line of the form

77	-	K	$\lambda+1$

to table of control transfers, prepare instruction for reading standard routine from magnetic drum (for operator 21).

Operator 16 — check to see whether the available space in block γ-3664 is adequate for the given standard routine; if it is, jump to operator 21, if not go to operator 17.

Operator 17 — see whether other standard routines are located in the block set aside for the standard routines; if there are, go to operator 18, if not (i.e., the given standard routine cannot fit into the entire block γ-3664 after an allowance has been made for the table of control transfers), then go to operator 20 (halt).

Operator 18 — "erase" block for standard routines. More accurately, change all third addresses to 3777 in the table of control transfers.

Operator 19 — exit from interpreter.

Operator 20 — halt.

Operator 21 — enter standard routine into operating memory.

Operator 22 — check to see whether standard routine has been correctly introduced; if so, go to operator 23; if not, jump to operator 20.

Operator 23 — prepare for modification of standard routine (form instruction for transmission and constant for determining end of standard-routine modification).

Operator 24 – read line from standard program.

Operator 25 – jump to operator 27 if the line read is a call for another standard routine, otherwise go to operator 26.

Operator 26 – modify line and place it back in standard routine.

Operator 27 – check to see whether modification of standard routine has been finished; if so, jump to operator 19, if not, go to operator 24.

Operator 28 (entrance to interpreter) – transfer control to operator 8 and jump to TsUK.

Interpreting Program. Occupies Positions 2.3222-2.3354 on Magnetic Drum (NO)

Адрес 1	Команда 2				№ оператора 3	4 Примечания
1	2				3	4
3645	34	–	3664	3651	1	5 Передача управления на оператор 4
6	33	3400	–	–	2	6 Останов при неверном вводе ИП
7	30	0402	3222	0132	3	7 Повторный ввод ИП
3650	31	–	3645	3675		
1	75	3675	3761	3646	4	8 Если ИП ввелась неверно, то останов – 2
2	26	3664	0026	a	5	$\boxed{-}\boxed{L+4}\boxed{-}\boxed{-}$
3	22	3654	a	3654		
4	17	3777	3774	a		9 При работе $\boxed{17}\boxed{L+3}\boxed{3774}\boxed{a}$
5	36	a	–	3657		10 Если $\gamma > 2000$, то – переход к оператору 7
6	17	3646	3774	a	6	3400 ⇒ γ
7	22	3711	a	3711	7	
3660	65	3711	0057	a1		$\boxed{16}\boxed{\gamma}\boxed{-}\boxed{r5}$
1	22	3775	a	a2		
2	26	a	0126	a		$\boxed{-}\boxed{-}\boxed{-}\boxed{\gamma}$
3	22	a2	a	a2		$\boxed{16}\boxed{\gamma}\boxed{3764}\boxed{\gamma}$
4	56	1520	2307	3502		11 Дополнение до контрольной суммы. При работе $\boxed{34}\boxed{-}\boxed{-}\boxed{L+4}$
5	22	3735	3761	3735	8	$\boxed{34}\boxed{-}\boxed{-}\boxed{x}$
6	26	3735	0026	r1		
7	22	r1	3714	3670		
3670	–	–	–	–		12 При работе $\boxed{-}\boxed{x}\boxed{r7}\boxed{r2}$ $\boxed{77}\boxed{-}\boxed{N}\boxed{-}$
1	17	r2	3762	r3		
2	65	3711	0057	3675		
3	26	3675	0126	r4	9	$\boxed{-}\boxed{-}\boxed{0016}\boxed{\gamma+i}$
4	35	3675	a1	3705		13 Если дошли до конца ТПУ, то – переход к оператору 13
3675	–	–	–	–	10	14 При работе $\boxed{16}\boxed{\gamma+i}\boxed{-}\boxed{r5}$
6	22	3675	3763	3675		15 Переадресация $\boxed{77}\boxed{-}\boxed{N'}\boxed{-}$
7	17	r5	3762	r6		
3700	75	r3	r6	3673		16 Если номера СП не совпадают, то – переход к оператору 9
1	17	r5	3764	r6	11	$\boxed{77}\boxed{-}\boxed{-}\boxed{K+1}$
2	35	r6	3764	3712		17 Если $K+1 = 3777_8$, то – переход к оператору 15
3	77	–	–	3706	12	18 Выполняет на МУКе оператор 14
4	34	–	–	3735		19 Передача управления на выход из ИП
5	22	a1	3763	a1	13	20 Увеличивает верхнюю границу ТПУ
6	20	r4	3765	r6	14	$\boxed{-}\boxed{-}\boxed{-}\boxed{\gamma+i}$
7	22	3766	3735	3710		
3710	–	–	–	–		21 При работе $\boxed{20}\boxed{r2}\boxed{r6}\boxed{x}$ 22 Изменение строки x в программе
1	37	–	–	r5		23 При работе $\boxed{37}\boxed{\gamma}\boxed{-}\boxed{r5}$
2	22	3767	r3	3713	15	
3	–	–	–	–		24 При работе $\boxed{30}\boxed{0402}\boxed{2000+N}\boxed{-}$
4	31	–	r7	r2		25 Выборка строки из ТСП $\boxed{30}\boxed{-}\boxed{-}\boxed{\lambda+1}$
5	17	r7	3764	r6		$\boxed{-}\boxed{-}\boxed{-}\boxed{K}$
6	62	a3	r6	r8		$\boxed{-}\boxed{-}\boxed{-}\boxed{K+1}$
7	62	r8	3761	a3		
3720	22	3770	r4	3721		
1	–	–	–	–		26 При работе $\boxed{22}\boxed{r3}\boxed{a3}\boxed{\gamma+i}$ 27 Засылки строки в ТПУ
2	42	r8	3763	r8		$\boxed{-}\boxed{-}\boxed{-}\boxed{K-2000}$
3	26	r8	0013	r5		$\boxed{-}\boxed{-}\boxed{K-2000}\boxed{-}$
4	22	3771	r5	3740		
5	66	a1	0113	r1	16	$\boxed{-}\boxed{2000}\boxed{ТПУ_к+1-2000}\boxed{-}$

Interpreting Program. Occupies Positions 2.3222-2.3354 on Magnetic drum (NO) (continued)

1	2				3	4
3726	76	r1	r5	3737		28 Если для СП хватает свободного места, то – переход к оператору 21
7	35	в3	3776	3736	17	29 Если (в3) = (3776), то СП не уменьшится на всем массиве ТПУ$_к$ + 1 ÷ 3664
3730	–	в2	–	3731	18	
1	–	–	–	3665		30 При работе $\boxed{16}\boxed{\gamma+j}\boxed{3764}\boxed{\gamma+j}$
2	22	3731	3736	3731		31 Переадресация
3	76	3731	в1	3731		32 Повторение цикла, если очищена еще не вся ТПУ
4	–	3776	–	в3		
5	–	–	–	–	19	33 При работе $\boxed{34}\boxed{-}\boxed{-}\boxed{\chi}$
6	33	0001	–	0001	20	
7	–	–	–	–	21	33 При работе $\boxed{30}\boxed{\varepsilon_{сп}}\boxed{\lambda_{сп}}\boxed{\lambda+1}$
3740	–	–	–	–		33 При работе $\boxed{31}\boxed{-}\boxed{\lambda-1}\boxed{r3}$
1	75	r3	3761	3736	22	34 Если СП ввелись неверно, то – останов
2	22	3770	в3	3755	23	35 Образование команды засылки
3	22	3772	r5	3744		
4	–	–	–	–		36 При работе $\boxed{22}\boxed{3755}\boxed{\lambda}\boxed{r5}$
5	66	3755	0026	r6	24	
6	22	r6	3771	3747		37 Образование команды выборки из СП
7	–	–	–	–		38 Выборка строки из СП 39 При работе $\boxed{-}\boxed{\lambda+i}\boxed{1777}\boxed{r3}$
3750	17	r3	3764	r7	25	
1	35	r7	3764	3756		40 Если строка является обращением к СП, то – переход к оператору 27
2	17	r3	3773	r7	26	
3	26	r7	0112	r7		
4	51	r7	r8	r2		

1	2				3	4
3755	–	–	–	–		41 При работе $\boxed{22}\boxed{r3}\boxed{r2}\boxed{\lambda+i}$ 42 Засылка в СП
6	62	3755	3761	3755	27	43 Переадресация
7	75	3755	r5	3745		44 Если обработана не вся СП, то – переход к оператору 24
3760	34	–	–	3735		45 Передача управления на выход из ИП
1	77	3777	3777	3777		46 В ячейках 3761 ÷ 3773 и 3776 находятся константы
2	77	–	3777	–		
3	64	0001	–	–		
4	77	–	–	3777		
5	–	–	0016	3777		
6	20	r2	r6	–		
7	30	0402	2000	–		
3770	22	r3	r2 = в3–16	–		
1	31	–	1777	r3		
2	22	3755	2000	r5		
3	–	2000	2000	2000		
4	–	3777	–	–		
5	16	–	3764	–		
6	–	–	–	3665		47 Верхняя граница массива для СП плюс 1
7	34	–	3735	3665	28	43 Передача управления на оператор 8

Interpreting Program. Occupies Positions 2.3222-2.3354 on Magnetic Drum (NO)

Callout Key

1) Address; 2) instruction; 3) number of operator; 4) comments; 5) transfer control to operator 4; 6) halt where IP is incorrectly introduced; 7) repeat entrance to IP; 8) if IP is incorrectly introduced, then halt – 2; 9) for operation; 10) if $\gamma > 2000$, jump to operator 7; 11) complement to check sum. For operation; 12) for operation; 13) if the end of the TPU has been reached, jump to operator 13; 14) for operation; 15) indexing; 16) if numbers of SP do not agree, jump to operator 9; 17) if $K + 1 = 3777$, jump to operator 15; 18) execute operator 14 at MUK; 19) transfer control to exit from IP; 20) increase upper limit of TPU; 21) for operation; 22) change line χ in program; 23) for operation; 24) for operation; 25) read line from TSP; 26) for operation; 27) place line into TPU; 28) if available area is sufficient for SP, jump to operator 21; 29) if $(s3) = (3776)$, SP will not fit into entire lot $TPU_K + 1$-3664; 30) for operation; 31) indexing; 32) repeat cycle if the entire TPU has not been erased; 33) for operation; 34) if SP has been improperly introduced, halt; 35) formation of transmission commands; 36) for operation; 37) formation of instruction to read from SP; 38) read line from SP; 39) for operation; 40) if line is a call for an SP, jump to operator 27; 41) for operation; 42) place into SP; 43) indexing; 44) if entire SP has not been processed, jump to operator 24; 45) transfer control to exit from IP; 46) locate constants in locations 3761-3773 and 3776; 47) upper limit of block for SP plus 1; 48) transfer control to operator 8.

Notes: IP) interpreting program; TPU) table of control transfer; TPU_K) end of table of control transfer; SP) standard routine; TSP) table of standard routines.

Running Contents of Certain Working Locations

s1	16	TPU_K+1	-	r5
s2	16	γ	3764	γ
s3	-	-	-	K+1

Note: TPU_K) End of table of control transfers; K + 1) beginning of standard routine introduced by the latter into the operating memory.

Absolute Addresses Assigned to Working Locations of Interpreter

r1	3755
r2	3713
r3	3670
r4	3747
r5	3744
r6	3740

r7	3737
r8	3710
s	3645
s1	3774
s2	3775
s3	3731

Appendix 1

Call for Compiling Program

At the beginning of the execution of the main program, it is necessary to execute the following instructions

$L + 1$	30	0402	3001	0018
$L + 2$	31	-	0001	0005
$L + 3$	77	α_H	α_K	0001
$L + 4$	n	β_H	β_K	A

Here α_N, α_K are the beginning and end of the main program (for that portion of it which contains all calls for standard routines);

β_N, β_K are the beginning and end of the block set aside for standard routines;

<u>n</u> is the number of standard routines loaded from punched cards;

A is the address of the first location in block A-A + 0157 set aside for the compiler.

Locations with addresses from 0001 to 0017 are working locations.

Appendix 2

Call for Interpreting Program

Before execution of the main program it is necessary to execute the following instructions

$L + 1$	30	0402	3222	0132
$L + 2$	31	-	3645	3675
$L + 3$	77	γ	-	3645

Here γ is the beginning of block γ-3777, set aside for standard routines and the interpreting program.

Appendix 3

Table of Blocks in Compiler

Адрес останова 1	Содержимое БЗК 2	Содержимое Б32П 3	Причина останова 4	Что делать 5
0003	$\boxed{33}\boxed{-}\boxed{-}\boxed{0003}$	00	6 Неверно считалась с МБ первая часть КП	7 Пуск на автомате или на циклах (повторится ввод)
4+0112	$\boxed{33}\boxed{0001}\boxed{-}\boxed{0003}$	77	8 Неверно считалась с МБ основная часть КП	9 Пуск на автомате (повторится ввод)
4+0071	$\boxed{33}\boxed{-}\boxed{-}\boxed{0001}$	77	10 Неверно ввелась СП	11 Повторить ввод; если СП вводилась с перфокарт, то переложить соответствующий комплект перфокарт; пуск на автомате или на циклах
4+0071	$\boxed{33}\boxed{-}\boxed{-}\boxed{0001}$	00	12 На массиве $\beta_н \div \beta_к$ не помещаются все СП	13 Произвести соответствующее исправление в программе (в командах обращения к КП).
4+0071	$\boxed{33}\boxed{-}\boxed{-}\boxed{0001}$	22	14 В основной программе используется больше 22 СП (в десятичной системе)	15 КП применить нельзя
4+0102	$\boxed{33}\boxed{3777}\boxed{-}\boxed{-}$	~	16 Не указан номер СП в команде обращения к СП. В первом адресе ячейки 0014 указано место этого обращения к СП	17 Произвести соответствующие исправления в программе
4+0103	$\boxed{31}\boxed{-}\boxed{\sim}\boxed{0010}$ или $\boxed{31}\boxed{0001}\boxed{\sim}\boxed{0010}$	~	18 Неверно указан номер СП в команде обращения к СП. В первом адресе ячейки 0014 указано место неверного обращения к СП	19 Произвести соответствующее исправление в программе

Appendix 3 (continued)

Callout Key

1) Address of halt; 2) contents of BZK; 3) contents of BZ2P; 4) reason for halt; 5) what is to be done; 6) improper readout from MB of first portion of KP; 7) automatic restart or start with looping (repeat loading); 8) improper reading of main portion of KP from MB; 9) restart in automatic operation (repeat loading; 10) SP improperly introduced; 11) repeat loading: if SP is loaded from punched cards, then replace corresponding pack of cards; start in automatic operation, or with looping; 12) entire SP will not fit into block β_N-β_K; 13) make appropriate correction in program (in instructions calling for KP); 14) more than 22 SP are used in main program (in decimal system); 15) KP cannot be used; 16) number of SP is not shown in instruction calling for SP. In first address of location 0014 the location of this call for the SP is shown; 17) m.ke appropriate correction to program; 18) number of SP is incorrectly shown in instruction calling for SP. In the first address of location 0014 the location of the incorrect call for the SP is shown; 19) make appropriate correction to program.

Notes: MB) magnetic drum; KP) compiling program; SP) standard routine.

Appendix 4

Table of Blocks in Interpreting Program

Адрес 1 останова	Содержимое 2 БЗК	Содержимое БЗ2П 3	Причина останова 4	Что делать 5
3646	$\boxed{33}\boxed{3400}\boxed{-}\boxed{-}$	77	6 Неверно ввелась ИП	7 Пуск на автомате или на циклах (повторится ввод)
3736	$\boxed{33}\boxed{0001}\boxed{-}\boxed{0001}$	77	8 Неверно ввелась СП	9 Повторить ввод: пуск на автомате или на циклах
3736	$\boxed{33}\boxed{0001}\boxed{-}\boxed{0001}$	00	10 Не выделенном массиве не помещается самая длинная СП. В третьем адресе ячейки 3735 указано место обращения к этой СП, во втором адресе ячейки 3670 находится номер этой СП	11 Произвести соответствующее исправление в программе (в командах обращения к ИП)
3737	$\boxed{33}\boxed{3777}\boxed{-}\boxed{-}$		12 Не указан номер СП в команде обращения к СП. В третьем адресе ячейки 3735 указано место этого обращения к СП	13 Произвести соответствующее исправление в программе
3740	$\boxed{31}\boxed{-}\boxed{\sim}\boxed{3670}$ $\boxed{31}\boxed{0001}\boxed{\sim}\boxed{3670}$ $\boxed{31}\boxed{3777}\boxed{\sim}\boxed{3670}$		14 Неверно указан номер СП в команде обращения к СП. В третьем адресе ячейки 3735 указано место неверного обращения к СП	15 Произвести соответствующее исправление в программе

1) Halt address; 2) contents of BZK; 3) contents of BZ2P; 4) reason for halt; 5) what is to be done; 6) IP incorrectly introduced; 7) restart in automatic operation or with looping (repeated loading); 8) SP improperly introduced; 9) repeat loading: restart in automatic operation or with looping; 10) the longest SP will not fit into block set aside for it. In third address of location 3735 the location of the call for this SP is shown, while the second address of location 3670 contains the number of this SP; 11) make appropriate correction to program (in instructions: IP); 12) number of the SP is not shown in instruction calling for SP. Third address of location 3735 shows the location of this call for the SP; 13) make appropriate correction to program; 14) number of SP is shown incorrectly in instruction calling for SP. The third address of location 3735 shows the location of the incorrect call for the SP; 15) make appropriate correction to program.

Note: IP) Interpreting program; SP) standard routine.

Appendix 5

Replacing Standard Routines

In working with the compiler it is possible to make simple substitutions of standard routines available in the library and already written on magnetic drum or magnetic tape where other routines are more desirable for various considerations from the viewpoint of the programmer. To do this, it is sufficient to include the new standard routine among the standard routines loaded from punched cards, giving it the number of the old standard routine, and making no other changes in the main program. The new standard routine should, of course, satisfy all of the conditions that were met when the standard subroutines were written (see §§ 7 and 8).

Manu-script Page No.	[List of Transliterated Symbols]
1	ОПМ = OPM = Otdel Prikladnoy Matematiki = Division of Applied Mathematics
7	Н = N = Nachalo = beginning
7	К = K = Konets = end
13	БЗК = BZK = blok zapominaniya komand = instruction storage unit
13	БЗ 2П = BZ2P = blok zapominaniya poryadka vtorogo chisla = = second-number order storage unit
14	ЦУК = TsUK = [not identified]
19	П = P = programma = program
32	ТН = TN = tablitsa nachal = table of initial locations
32	КМ = KM = konets massiva = end of block set
32	л = l = lenta = tape
40	МУК = MUK = [not identified]
44	СП = SP = standartnaya programma = standard program
44	ИП = IP = interpretiruyushchaya programma = interpreting program
44	ТПУ = TPU = tablitsa peredach upravleniya = table of control transfer
45	ЩП = TSP = tablitsa standartnykh programm = table of standard routines
49	КП = KP = kompiliruyushchaya programma = compiling program
49	МБ = MB = magnitnyy baraban = magnetic drum

DISTRIBUTION LIST

DEPARTMENT OF DEFENSE	Nr. Copies	MAJOR AIR COMMANDS	Nr. Copies
		AFSC	
		SCFDD	1
		DDC	25
		TDBTL	5
		TDBDP	5
HEADQUARTERS USAF		SSD (SSF)	2
AFCIN-3D2	1	APGC (PGF)	1
ARL (ARB)	1	ESD (ESY)	1
		RADC (RAY)	1
		AFWL (WLF)	1
		AFMTC (MTW)	1
OTHER AGENCIES		ASD (ASYIM)	2
CIA	1		
NSA	6		
DIA	9		
AID	2		
OTS	2		
AEC	2		
PWS	1		
NASA	1		
ARMY (FSTC)	3		
NAVY	3		
NAFEC	1		
RAND	1		
SPECTRUM	1		
AFCRL (CRXLR)	1		